$$(Skew/NoD) * NoV * AA / [1 - Reserved\_Factor - (Skew/NoD) * (\Sigma_{i=1}^{Nov} P_i)/ TR] \leq$$

$$T \leq (1- Reserved\_Factor) * B_{max} / [(1 - B\_Save) * (\Sigma_{i=1}^{Nov} P_i)] \qquad (10)$$

For multiple storage device case under substantially unbalanced conditions:

$$MaxNoV\_perDevice * AA / [1 - Reserved\_Factor - MaxAggRate\_perDevice / TR] \leq$$

$$T \leq (1- Reserved\_Factor) * B_{max} / [(1 - B\_Save) * (\Sigma_{i=1}^{Nov} P_i)] \qquad (11)$$

In the practice of the disclosed methods and systems, Resource Model Equations (9), (10) and (11) may be employed for I/O admission control and the read-ahead estimation in a manner similar to that previously described for Resource Model Equations (4), (7) and (8A).

Resource Modeling: Read-ahead size Calculation

In another embodiment of the disclosed methods and systems, read-ahead size may also or alternatively be determined in addition to making admission control decisions and cycle time determinations. Read-ahead size may be so determined in one exemplary embodiment based on the previously described relationship given in equation (1). In this regard, equation (1) may be re-written to derive the number of read-ahead blocks for each viewer N_i given the block size BL, the estimated consumption rate P_i and the calculated cycle T, as follows:

$$N_i = T * P_i / BL \qquad (12)$$

The calculated number of read-ahead blocks N_i may not always be an integer number, and may be adjusted to an integer number using any desired methodology suitable for deriving an integer number based on a calculated non-integer value, *e.g.,* by rounding up or rounding

down to the nearest integer value. In one exemplary embodiment, read-ahead size may be implemented based on a calculated value of N_i by alternately retrieving the largest integer less than the calculated N_i, and the smallest integer larger than the calculated N_i, in different (*e.g.*, successively alternating) cycles. In this way, it is possible to retrieve an average amount of data for each viewer that is equivalent to N_i, while at the same time enabling or ensuring continuous playback.

Resource Modeling for Multiple Buffering Implementations

Embodiments of the disclosed methods and systems may also be implemented in a manner that is capable of handling unpredictable playback dynamics, *i.e.,* to address actual I/O activities. In this regard, buffer space may be allocated in a manner that is event driven and that reflects the state of each viewer at any instantaneous moment. For example, in one exemplary embodiment, a sliding window buffer approach that includes multiple buffers may be implemented. One example of a sliding window buffer approach using two buffers is illustrated in FIG. 2. As shown in FIG. 2, at time 0 a first buffer space of N (read-ahead) blocks is allocated as the "D_1" buffer" to fetch data from a storage device (*e.g.,* a storage disk), and the D_1 buffer is filled before T_1 (cycle) seconds expire. After the first buffer space has been filled and becomes ready for being sent it is denoted in FIG. 2 as the "B_1" buffer. In a transient fraction of time ("δT_1"), sending of data in the B_1 buffer starts the first buffer space becomes a sent "S_1" buffer as illustrated in FIG. 2. Simultaneously with sending data in the B_1 buffer and with change of the B_1 buffer to the D_1 buffer, a second buffer space is allocated as the "D_2" buffer to fetch data in the second cycle T_2 and is filled and sent in the same manner as the first buffer space in the first cycle. The I/O cycles continue sequentially in the same way as shown in FIG. 2.

Due to the event driven nature of the double buffering embodiment of FIG. 2, unexpected changes in consumption may not substantially change buffer space requirements. If a given viewer experiences network congestion, for example, then the time it takes to transmit the S_i buffer will take longer than T_i - δT_i, meaning that the next D_{i+1} buffer will transition into a B_{i+1}

buffer and stay there for sufficient time until the S_i buffer is transmitted. However, space for the D_{i+2} buffer will not be allocated until the S_i buffer is completely sent. Therefore, using this exemplary embodiment, buffer space consumption may remain substantially stable in response to unexpected system behaviors.

Those embodiments employing multiple buffers require increased buffer space, *e.g.,* use of double buffering serves to double the buffer space requirement. As a consequence, the disclosed resource model methodology (*e.g.,* any given one of the previously described Resource Model Equations) may be modified so that various buffering implementation schemes (and their impacts on actual buffer consumption) may be reflected. For example, in one exemplary embodiment, a buffer memory parameter ("Buffer_Multiplcity") to reflect characteristics of implemented buffering techniques and their implicit buffering requirement alteration. In the case of multiple buffer implementation, value of a Buffer_Multiplcity factor may be set after characteristics of a multiple buffer implementation are decided upon. As just one example, the buffer memory parameter Buffer_Multiplcity may be employed to represent multiple buffering by modifying respective Resource Model Equations (9), (10) and (11) as follows:

For single storage device case:

$$NoV * AA / [1 - Reserved\_Factor - (\Sigma_{i=1}^{Nov} P_i)/ TR] \leq T$$

$$\leq (1- Reserved\_Factor)*B_{max} / \{Buffer\_Multiplcity*[(1 - B\_Save)*(\Sigma_{i=1}^{Nov} P_i)]\} \qquad (13)$$

For multiple storage device case under substantially balanced conditions:

$$(Skew/NoD) *NoV * AA / [1- Reserved\_Factor - (Skew/NoD) * (\Sigma_{i=1}^{Nov} P_i)/ TR] \leq T \leq (1- Reserved\_Factor)*B_{max} / \{Buffer\_Multiplcity*[(1 - B\_Save) * (\Sigma_{i=1}^{Nov} P_i)]\} \qquad (14)$$

For multiple storage device case under substantially unbalanced conditions: